Integrating HLA, SCORM, and Instruction: Three Prototypes

Philip V. W. Dodds
ADL Chief Architect

J. D. Fletcher
Institute for Defense Analyses

ABSTRACT: *Examination of three ADL prototypes, two of which integrate the High Level Architecture (HLA), the Sharable Content Object Reference Model (SCORM), and training applications suggests that four basic components are needed: a communications link, access to simulation state data, a means to map student performance to simulation state data, and a means to map simulation states and student performance to learning objectives. In two of the prototypes, the HLA Run Time Interface (RTI) provides a communications link between an HLA compliant simulation and a SCORM Learning Management System (LMS). The third prototype uses a commercial off-the-shelf game platform for this link. All three prototypes use SCORM content objects to transmit simulation and student state data to an LMS, which then identifies simulation scenarios needed to achieve student learning objectives. A more standardized object-oriented approach with a reusable HLA-SCORM interface object could be developed to both encapsulate complex data constructs and transmit state data between the LMS and the simulation environment.*

1. Simulation and Advanced Distributed Learning

However carefully we plan our military operations, something, in fact many things, will go wrong. Exigencies inevitably arise that we did not anticipate, for which we were not prepared, and to which we must respond as, literally, a matter of life and death. As a consequence, our training simulations must reflect the fluidity of modern operational environments, adapting on demand and in real time to the needs and responses of their users. They also must be accessible as needed in garrison, in transit to operational trouble spots, and in the field. In short they need to be available anytime, anywhere.

These requirements for adaptability and accessibility match the well-noted goals of the Advanced Distributed Learning (ADL) initiative [1]. These goals are to ensure the availability of education, training, and performance aiding materials that are tailored to user requirements and accessible anytime, anywhere. Support for rapidly adaptive simulation is, then, a primary component of the ADL initiative. However, these simulations have to start somewhere, with something. In ADL, they begin with sharable learning objects.

The ADL initiative is preparing for a future in which communication networks and personal delivery devices are pervasive, inexpensive, and effectively transparent to users through ease of use, expanded bandwidth, and portability. It will establish knowledge libraries, or repositories, where learning 'objects' may be accumulated and cataloged for broad distribution and use. Because of their enhanced accessibility, these objects will be ready for assembly on demand and in real time into instructional and performance aiding materials that are tailored to the capabilities, intentions, and learning state of each individual or group of individuals needing them.

2. Learning Objects and SCORM

To date much of the ADL effort has been devoted to the specification of instructional objects that will populate learning libraries and other Web-available repositories. These objects are separated from context-specific run-time constraints and proprietary systems so that they can be incorporated into other applications. They have common interfaces and data exchange formats. They are accessible so that they can be indexed and readily found or "discovered"; interoperable so that they operate across a wide variety of hardware, operating systems, and Web browsers; durable so that they do not require modification as versions of the underlying software systems change; and reusable so that they can be adapted and used by many different development tools.

Specifications for these learning objects are provided by the Sharable Content Object Reference Model (SCORM). SCORM assumes a run-time environment in which these objects will be managed by middle-ware Learning Management Systems (LMSs). It is intended to ensure that any conforming LMS will be able to access conforming repositories of learning objects that have been authored by different developers using different tools, locate needed objects and launch them in executable applications, and provide data exchange among the objects [2]. SCORM has evolved through several versions, the most current of which (SCORM 2004) can be found at http://www.adlnet.org.

Many technicians, software engineers, instruction designers, and cognitive researchers from all economic sectors in the Americas, Europe, and Asia have participated in the development of SCORM. The task of specifying and developing sharable learning objects has become a global effort. The primary contribution of the ADL initiative has been to orchestrate this effort and document its results.

3. SCORM and HLA

If SCORM is to support the agile simulations envisioned by the ADL initiative, it must be harmonized with High Level Architecture (HLA) [3]. The bridge to be built between simulation, with HLA, on one hand, and ADL, with its SCORM specifications, on the other is both a technical and instructional challenge. While similar at a high level, HLA and SCORM are very different architectures. This difference is appropriate because they address different domains. HLA is intended to enable a multiplicity of processes to connect and transfer information, often in real time. SCORM is intended to support directed learning experiences that develop specific human competencies. SCORM can provide the means to track a learner's mastery in order to devise an appropriate program of instruction. HLA can provide the means to deliver simple or complex simulation experiences once such a program of instruction has been devised.

The challenge in blending these architectures is connecting the "teaching" and "doing" parts of learning in one seamless (to the learner) environment. It requires mapping specific skills and competencies to demonstrated results in a simulation environment.

SCORM, therefore, is key to the development, packaging, aggregation, and sequencing of instructional material as objects. Its specifications define how these objects can be tagged for later discovery in terms of descriptive characteristics such as targeted learning objectives. Simulations could similarly be developed that map to the same vocabulary of descriptors so that a learner, program developer, or computer algorithm could discover both instruction and simulation objects and retrieve them on demand. This paper discusses three prototype efforts that incorporate both HLA and SCORM to discover, retrieve, and present contextually relevant objects for instruction and simulation and then track a learner's performance and progress toward attaining instructional objectives.

4. Three Prototype Cases

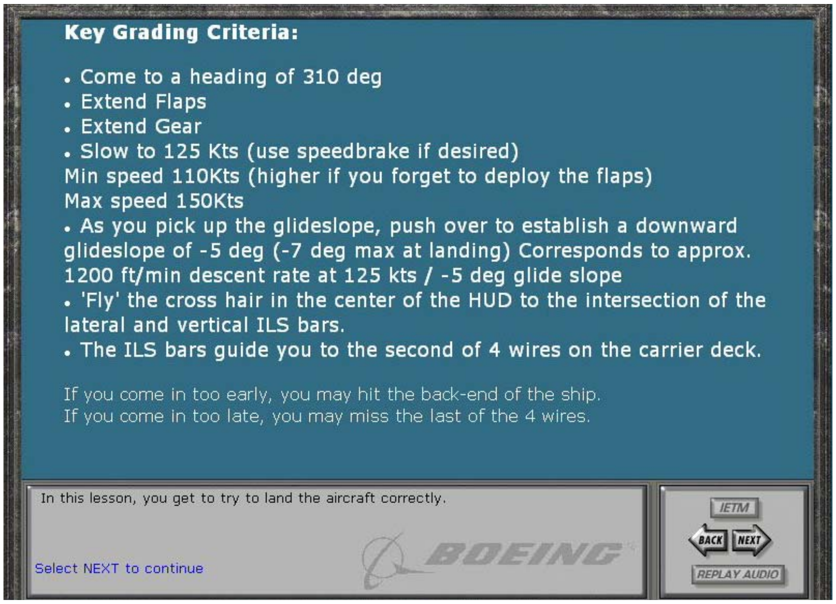
Three prototypes are now in development that connect a LMS to a simulation platform such that data may be exchanged between them. Two of the three prototypes connect to HLA-based simulation; the third uses a commercial off the shelf 3D game engine. All three examine methods for exchanging learner performance information from the simulation environment to the LMS so that the LMS can provide appropriate and relevant instruction. These prototypes build upon discussions of similar concepts between the Defense Modeling and Simulation Office (DMSO) and ADL as well as prototypes sponsored by DMSO in 2002.

4.1 Case 1: The Boeing Company's SCORM/HLA prototype

Early in the development of SCORM's sequencing and navigation work, ADL collaborated with Boeing to develop instructional use cases that would inform the new content sequencing technical specification then in development. Their use case prescribed specific rules and sequencing behaviors and described the launching of a simulation and the evaluation of learner performance that would subsequently affect the instructional path. During the development of the sequencing specification, it was decided that more research was required to determine how to integrate simulations in a learning management system. The prototype described here was developed as a result of this work.

The lead Engineer and Principal Investigator for R&D at Boeing's Training Software Systems in St. Louis created a prototype that links an LMS to an HLA-based simulation. This was an internal Boeing research project that leveraged Boeing's existing simulation and training capabilities, linking the two for the first time.

A "lesson" was defined as a collection of sharable content objects (SCOs), each addressing one facet of an instrument landing system (ILS). Another SCO acts as a "virtual flight instructor" who presents a specific scenario to the learner. The system then initializes a flight simulation that fits the scenario. The learner flies the approach while data from the simulation is fed back to the learning environment. During the simulation, the SCO (the virtual flight instructor) provides feedback such as "you are a little too high". When the landing is complete, end-state data is returned to the LMS for display to the learner and for the LMS to use in determining the student's mastery of various learning objectives.



Key Grading Criteria:

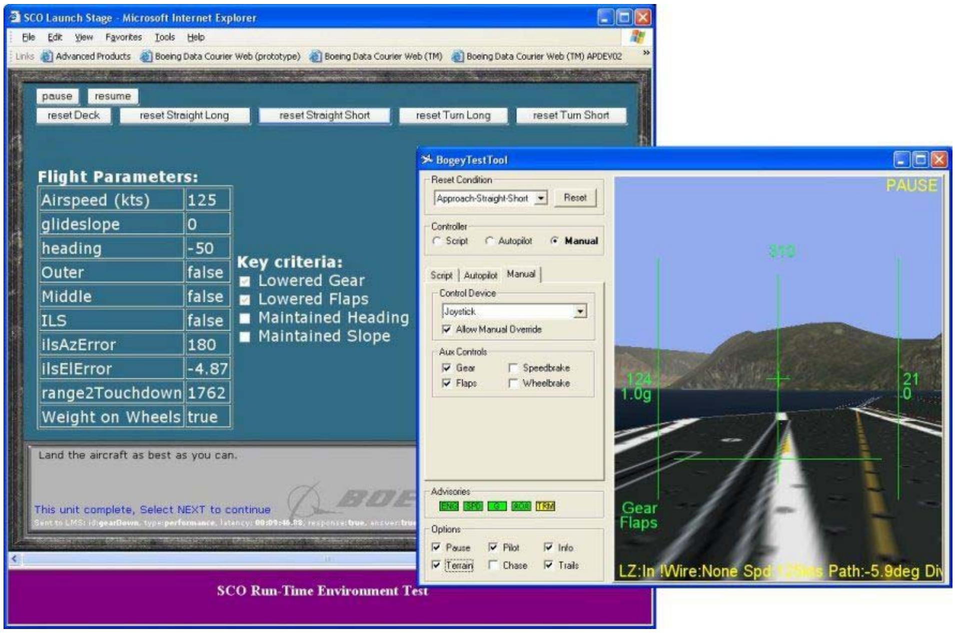
- Come to a heading of 310 deg
- Extend Flaps
- Extend Gear
- Slow to 125 Kts (use speedbrake if desired)
Min speed 110Kts (higher if you forget to deploy the flaps)
Max speed 150Kts
- As you pick up the glideslope, push over to establish a downward glideslope of -5 deg (-7 deg max at landing) Corresponds to approx. 1200 ft/min descent rate at 125 kts / -5 deg glide slope
- 'Fly' the cross hair in the center of the HUD to the intersection of the lateral and vertical ILS bars.
- The ILS bars guide you to the second of 4 wires on the carrier deck.

If you come in too early, you may hit the back-end of the ship.
If you come in too late, you may miss the last of the 4 wires.

In this lesson, you get to try to land the aircraft correctly.

Select NEXT to continue

Figure 4.1. ***Instructional "set up" prior to launching simulation***



SCO Launch Stage - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links Advanced Products Boeing Data Courier Web (prototype) Boeing Data Courier Web (TM) Boeing Data Courier Web (TM) APDEV02
pause
resume
reset Deck
reset Straight Long
reset Straight Short
reset Turn Long
reset Turn Short

Flight Parameters:

Airspeed (kts)	125
glideslope	0
heading	-50
Outer	false
Middle	false
ILS	false
ilsAzError	180
ilsElError	-4.87
range2Touchdown	1762
Weight on Wheels	true

Key criteria:

- ☑ Lowered Gear
- ☑ Lowered Flaps
- □ Maintained Heading
- □ Maintained Slope

Land the aircraft as best as you can.

This unit complete, Select NEXT to continue

Sent to LMS: id:gearDown, type:performance, latency: 00:09:46.88, response:true, answer:true

Reset Condition: Approach-Straight-Short

Controller: ○ Script ○ Autopilot ● Manual

Control Device: Joystick

☑ Allow Manual Override

Aux Controls: ☑ Gear □ Speedbrake ☑ Flaps □ Wheelbrake

Advisories: ENG SPD G ROB TRM

Options: ☑ Pause ☑ Pilot ☑ Info ☑ Terrain □ Chase ☑ Trails

LZ:In !Wire:None Spd:125kts Path:-5.9deg Di

Figure 4.2. ***The simulation feeding state data back to the LMS***

In general, the sequence followed is:

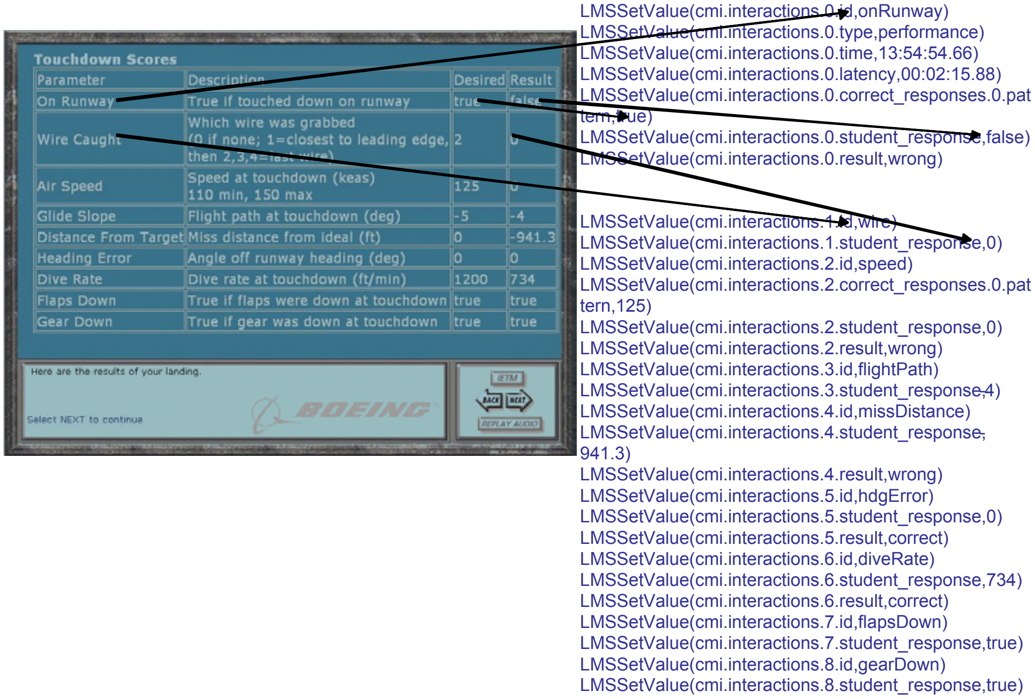
- The LMS determines the next SCO to present
- The LMS launches an ADL-conformant SCO
- The SCO presents lesson objective and lesson materials
- The SCO sends an HLA packet to initialize the simulation
- The Student lands the aircraft in the simulation
- The simulation sends aircraft information and student actions to the SCO using HLA
- The SCO evaluates student actions, displays feedback to student, sends information to the LMS using SCORM, and sends override commands to the simulation using the HLA provided Run Time Interface (RTI).
- The LMS remediates if necessary or determines next SCO…

Figure 4.1 shows this sequence in action as the SCO is used to present the scenario tasking to the student. Figure 4.2 shows the feedback provided back to the student as aircraft and student state information is carried back and forth between the ongoing simulation and the LMS by a SCO using an HLA RTI to effect the communication. Figure 4.3 shows some of the information exchanged between the simulation and the LMS. Finally, Figure 4.4 shows the mapping between simulation state data and LMS data elements performed by this system.

The Boeing prototype illustrates how simulation state data can be interpreted in terms of a specific instructional context and how that data can then be used to modify the path through the instructional material. These capabilities are achieved both through thoughtful analysis of simulation state data during the design of the instruction and by mapping those data to specific instructional content and sequencing rules. At run-time, data from the simulation is obtained from the HLA run time interface (RTI) and delivered back to the LMS so that it may fire rules such as "if not mastered, remediate this objective". The HLA simulation proved a useful environment because it can expose state data through the RTI in a standardized way. Notably, the design and analysis of data and mapping it to performance data is a non-trivial undertaking.

	Id	**Type**	**Time**	**Latency**	**Correct_ response pattern**	**Student_ response**	**Result**
0	**onRunway**	**performance**	**2:50:32 PM**	**01:57.6**	**TRUE**	**TRUE**	**CORRECT**
1	**wire**	**performance**	**2:50:32 PM**	**01:58.0**	**2**	**2**	**CORRECT**
2	**speed**	**performance**	**2:50:32 PM**	**01:58.4**	**125**	**1**	**WRONG**
3	**flightPath**	**performance**	**2:50:33 PM**	**01:58.9**	**-5**	**-6**	**CORRECT**
4	**missDistance**	**performance**	**2:50:33 PM**	**01:59.3**	**0**	**-3.1**	**CORRECT**
5	**hdgError**	**performance**	**2:50:34 PM**	**01:59.9**	**0**	**0**	**CORRECT**
6	**diveRate**	**performance**	**2:50:34 PM**	**0:02:00**	**1200**	**1311**	**CORRECT**
7	**flapsDown**	**performance**	**2:50:35 PM**	**0:02:01**	**TRUE**	**TRUE**	**CORRECT**
8	**gearDown**	**performance**	**2:50:36 PM**	**02:01:6**	**TRUE**	**TRUE**	**CORRECT**

Figure 4.3 Simulation state data obtained from the run time environment (RTI)



Touchdown Scores

Parameter	Description	Desired	Result
On Runway	True if touched down on runway	true	false
Wire Caught	Which wire was grabbed (0 if none; 1=closest to leading edge, then 2,3,4=last wire)	2	0
Air Speed	Speed at touchdown (keas) 110 min, 150 max	125	0
Glide Slope	Flight path at touchdown (deg)	-5	-4
Distance From Target	Miss distance from ideal (ft)	0	-941.3
Heading Error	Angle off runway heading (deg)	0	0
Dive Rate	Dive rate at touchdown (ft/min)	1200	734
Flaps Down	True if flaps were down at touchdown	true	true
Gear Down	True if gear was down at touchdown	true	true

Here are the results of your landing.

Select NEXT to continue

```
LMSSetValue(cmi.interactions.0.id,onRunway)
LMSSetValue(cmi.interactions.0.type,performance)
LMSSetValue(cmi.interactions.0.time,13:54:54.66)
LMSSetValue(cmi.interactions.0.latency,00:02:15.88)
LMSSetValue(cmi.interactions.0.correct_responses.0.pat
tern,true)
LMSSetValue(cmi.interactions.0.student_response,false)
LMSSetValue(cmi.interactions.0.result,wrong)

LMSSetValue(cmi.interactions.1.id,wire)
LMSSetValue(cmi.interactions.1.student_response,0)
LMSSetValue(cmi.interactions.2.id,speed)
LMSSetValue(cmi.interactions.2.correct_responses.0.pat
tern,125)
LMSSetValue(cmi.interactions.2.student_response,0)
LMSSetValue(cmi.interactions.2.result,wrong)
LMSSetValue(cmi.interactions.3.id,flightPath)
LMSSetValue(cmi.interactions.3.student_response,-4)
LMSSetValue(cmi.interactions.4.id,missDistance)
LMSSetValue(cmi.interactions.4.student_response,-
941.3)
LMSSetValue(cmi.interactions.4.result,wrong)
LMSSetValue(cmi.interactions.5.id,hdgError)
LMSSetValue(cmi.interactions.5.student_response,0)
LMSSetValue(cmi.interactions.5.result,correct)
LMSSetValue(cmi.interactions.6.id,diveRate)
LMSSetValue(cmi.interactions.6.student_response,734)
LMSSetValue(cmi.interactions.6.result,correct)
LMSSetValue(cmi.interactions.7.id,flapsDown)
LMSSetValue(cmi.interactions.7.student_response,true)
LMSSetValue(cmi.interactions.8.id,gearDown)
LMSSetValue(cmi.interactions.8.student_response,true)
```

Figure 4.4 Mapping simulation state data to LMS data elements

4.2 Case 2: SCORM/HLA prototype

A prototype funded through the ADL Joint CoLab in Orlando is underway to build a SCORM/HLA demonstration that integrates instruction and simulation. The work is being performed by the ADL Joint CoLab [4] and Intelligent Automation, Inc. [5] in Rockville Maryland. As of this writing, the project was still in the design stage.

The prototype incorporates an existing simulation that teaches Air Traffic Flow Coordinators to optimize air traffic flow through specific control spaces. The simulator is called the Collaborative Regional Flow Control (CRFC) Decision Support Tool (DST). Its design documents call for a prototype system that "elegantly incorporates didactic instruction, performance demonstration, guided simulation, performance-based assessment, and written assessment into a unitary learning experience."

The prototype will be managed by an LMS that launches a "simulation manager" content object (a SCO) that initializes and launches the HLA-based simulation. The SCO then becomes a federate of the simulation and provides a communications link between the federates and the LMS. The simulation includes a special client class that passes simulation messages to the SCO. One of these classes, called "SimAssessment", translates the messages from the simulation's RTI into SCORM performance data.

The method for mapping simulation data to LMS data is still in development. However the early design documents identify SCORM data elements to be used.

The prototype design includes an "RtiScoInterfaceAgent" that receives messages containing data about the state of the simulation. Thus there is the means to "listen in" to the data passing through the simulation's RTI, map certain data states to LMS performance metrics, and return these metrics back to the LMS.

The project is still in the design phase and the exact method of mapping HLA RTI data to LMS performance data has not been determined. However, the mechanics for connecting the two environments appear very similar to the Boeing prototype.

Function	LMS Data Model Elements	Simulation Purpose
Control (start/pause/restart)	cmi.suspend_data	Stores location and state data of the simulation between sim sessions
Setup Parameters	cmi.sttudent_data. max_time_allowed cmi.student_data. Mastery_score	Can be used to indicate the simulation termination conditions.
Personalization parameters	cmi.core.student._id cmi.core.student_name cmi.student_preference. Language cmi.student_preference. audio	Can be used to personalize simulation presentation.
Performance Data	cmi.core.score.raw cmi.core.score.max cmi.core.score.min cmi.objectives.n.score.raw cmi.objectives.n.score.max cmi.objectives.n.score.min	Native simulator performance metrics can be stored as a list of score elements. These could form part of an overall performance outcome measure.
Overall Status	cmi.core.lesson_status "passed, completed, failed, incomplete, not attempted"	Overall simulation status could be determined from the overall score. This outcome could then be used for student tracking and sequencing.

Figure 4.5 ***SCORM data elements to be used in the CRFC decision support tool.***

4.3 Case 3: Civil support team trainer prototype

Working with the Joint ADL Co-Lab, the Army Research, Development and Engineering Command, and the National Guard Bureau, Engineering & Computer Simulations, Inc. [6] (ECS) developed a training prototype that integrates simulation and SCORM-based learning content. Called the Civil Support Team Trainer (CCST), this system provides training exercises for individuals and groups in Civil Support Teams who must deal with weapons of mass destruction. The prototype provides short, goal-oriented simulation scenarios with specific training objectives.

An LMS is used to control the presentation of courseware from four different components: web-based instruction, virtual instruction, practice, and assessment. Web-based instruction was developed to explain a set of skills and how they are to be applied in a specific context. Once students complete the instruction, they enter a team-based simulation environment that builds proficiency for sub-tasks that contribute in turn to overall mastery for a given instructional objective. Assessments are gathered within the simulation scenarios by monitoring particular simulation state data, the results of which are returned to the LMS as objective "satisfied" or "not satisfied." The LMS then determines the appropriate instructional path based on mastery of the objective.

The simulations are created using a commercial off the shelf 3D simulation engine called Gamebryo from Numerical Design Limited (NDL) [7]. This engine is used by a number of game developers and is a simulation rendering platform that was integrated into the CCST prototype. Gamebryo is not HLA compliant and does not use the HLA RTI, but it has similar functionality to HLA-based simulations. It also has toolkits and interfaces that permit its data to be viewed and modified. The engine is aimed at developers of commercial games. There are many such simulation platforms/engines in the gaming industry that can be used to develop new "worlds." Each have their own strengths, weaknesses, capabilities and models. For this prototype ECS selected the Gamebryo engine as the simulation component of the overall training environment and then constructed "middleware" that operates between Gamebryo and the LMS to translate the state data of the simulation to mastery data the LMS can then comprehend and act upon.

A key to the success of this prototype was the concept of "chunking" simulation scenarios into a size and scope that address particular instructional objectives. This way simulation scenarios and instructional material can share performance data more precisely since both the scenario and the instruction were designed with the same objective in mind. This approach deviates from traditional simulators that have no overt instructional oversight or intervention.

5. Conclusions and Observations

All three prototypes use an LMS to manage the instruction. In other words, the LMS is "in charge". The LMS administers assessments, instructional content, and then launches a SCO that initializes and launches (or federates with) a simulation. When the simulation ends, the LMS in all three cases receives mastery information derived from simulation state-data. This approach of integrating two widely different environments, each with totally different data models (one for tracking learner performance, and one that models the world), appears to be a viable strategy for integrating "learning and doing".

All three prototypes create links between the LMS and the simulation by launching a SCO which then either connects to (via middleware) or becomes part of a simulation as a communications and translation bridge. The actual translation of simulation data occurs along this link so that each of the environments operate independently of one another. This means that the LMS and the simulation environments did not have to be modified to integrate and operate together.

Each of the prototype cases use scenarios that map to particular learning objectives. In Cases 1 and 2 (landing a plane and ILS training), the scenario initializes special-built simulations to create conditions that are specifically relevant for the given instructional objectives. In Case 3 (Civil Support Team Trainer), a simulated world is created and initialized to match the objectives. All three cases operate similarly except that Case 3 builds scenarios ground-up expressly to integrate with instructional material whereas the first two cases adapt pre-existing simulators.

All three cases report mastery information from the simulation back to the LMS by mapping simulation state to the LMSs data model. This process may well be the hardest part about integrating the environments. In Boeing's case, for example, mapping all of the elements of a successful landing requires keeping track of a large number of details such as speed, flight path, flaps position, landing gear state, etc. Designing a large number of scenarios could become complex and since a fair amount of time must be invested in developing instructional strategies that use this information.

These prototypes show that HLA provides a useful mechanism for transmitting simulation data through the RTI. SCOs are used as intermediate agents for hiding the complex data constructs that constitute proficiency in a simulation from the LMS. This process suggests a more standardized object-oriented approach might be developed for hiding the methods of mapping but exposes appropriate data forms to simulations and LMSs. One might devise a standardized HLA-SCORM interface object that could then be reused and reinitialized for each new scenario. HLA's RTI seems especially well suited for building such an object through the "federation object model" (FOM) process. This may be a good next step for development.

The three prototypes all must initialize simulations and save their state for later entry and use. Currently SCORM lacks an adequate mechanism to do this. However, the IMS Global Learning Consortium [8] may address this issue with the Sharable State Persistence (SSP) specification. SSP defines how an LMS can save and retrieve blocks of data that can then be used to initialize simulations. Best practices for implementing SSP are expected to appear by the end of 2004.

6. Summary

These prototypes suggest that to link learning environments to simulations, four basic components are required: a communications link, access to simulation state data, a map of simulation data to LMS performance data, and simulation scenarios that can map to learning objectives. Developing intermediate agents to aid in the mapping seems to be an area ripe for new research.

Going further, one can imagine agents becoming more intelligent and embedded within a simulation, intervening and adapting seamlessly with the simulation. The lessons learned in these prototypes lead us in this direction by creating direct links with data that help meet proficiency and instructional objectives. As agent technology improves, the functionalities of an LMS may migrate inside the simulation model and become part of it. We might then have the agile "simulation tutors" required by today's operating environments.

7. References

[1] Advanced Distributed Learning Initiative: www.adlnet.org

[2] Dodds, P. V. W. (Ed.) (2002) Sharable Courseware Object Reference Model (SCORM) Version 1.2 (IDA Document D-2677). Alexandria, VA: Institute for

Defense Analyses. Evolving versions of SCORM are posted on line at http://www.idanet.org.

[3] High Level Architecture: Defense Modeling and Simulation Office: https://www.dmso.mil/public/transition/hla/

[4] Joint ADL CoLab (Orlando, Florida): http://www.jointadlcolab.org/

[5] Intelligent Automation, Inc.: http://www.i-a-i.com/

[6] Engineering & Computer Simulations, Inc.: http://ecsorl.com/

[7] Gamebryo game engine from NDL: http://www.ndl.com/

[8] IMS Global Learning: http://www.imsglobal.org/

Author Biographies

PHILIP V. W. DODDS is a Project Analyst with Randall House Associates, Inc., Adjunct Staff Member with the Institute for Defense Analyses, and the Chief Architect of the Department of Defense Advanced Distributed Learning Initiative along with its Sharable Content Object Reference Model.

J. D. FLETCHER is a Research Staff Member in the Science and Technology Division of the Institute for Defense Analyses where he specializes in manpower, personnel, training, and human factors issues. He is the task leader for Advanced Distributed Learning tasks.